UNCLASSIFIED







(2)

# DSTO

## High Frequency Radar Division



**TECHNICAL REPORT**
**SRL-0131-TR**

# A GENERAL PURPOSE IONOSPHERIC RAY TRACING PROCEDURE

by

C. J. Coleman





APPROVED FOR PUBLIC RELEASE

UNCLASSIFIED

SURVEILLANCE RESEARCH LABORATORY

AR-008-171

DSTO AUSTRALIA

SURVEILLANCE RESEARCH LABORATORY



# High Frequency Radar Division



**TECHNICAL REPORT**
**SRL-0131-TR**

## A GENERAL PURPOSE IONOSPHERIC RAY TRACING PROCEDURE

by

C. J. Coleman

**SUMMARY**

A general purpose ionospheric ray tracing procedure is developed. The procedure is based on a numerical solution to the Haselgrove ray tracing equations and includes calculations of ionospherically induced Doppler shift. The procedure can handle a wide variety of ionospheric descriptions and includes the effect of the Earth's magnetic field.



AUG 93

APPROVED FOR PUBLIC RELEASE

POSTAL ADDRESS: Director, Surveillance Research Laboratory, PO Box 1500, Salisbury, South Australia, 5108

# CONTENTS



# APPENDICES

THIS IS A BLANK PAGE

# 1 INTRODUCTION

Radiowave systems that rely on the ionosphere have a need for reliable predictions of propagation performance. Unfortunately, the ionosphere has an extremely complex morphology that is continually changing and so these predictions are by no means easy. The ionospheric structure can be derived from soundings and models, but the effect on propagation can only be gauged from solutions to the appropriate electromagnetic field equations. For HF radiowaves, however, a geometric optics approximation is often adequate and so the propagation can be studied through a suitable ray tracing procedure. Such a procedure needs to take account of the Earth's magnetic field and to accept a variety of ionospheric descriptions. In addition, it will need to calculate quantities such as phase path, group path and ionospheric Doppler shift. It is the purpose of the current report to describe a computer procedure that can meet these demands.

Section 2 of this report presents the theoretical results upon which the procedure is based, Section 3 outlines the numerical techniques that are employed and Section 4 discusses the use of the procedure. The computer implementation is contained in Appendix A.

# 2 THEORY

The core of the procedure is based on a numerical solution to the Haselgrove ray tracing equations (Haselgrove, 1955, 1960 and 1963). These ordinary differential equations (ODE's) are given by

$$\frac{dx_i}{d\tau} = Ju_i - KYv_i \tag{1}$$

and

$$\frac{du_i}{d\tau} = L\frac{\partial X}{\partial x_i} + \sum_j (Ku_j + MYv_j)\frac{\partial (Yv_j)}{\partial x_i} \tag{2}$$

where $X=8.06\times10^{-5}N/f^2$, $f$ is the wave frequency ($MHz$), $N$ is the electron density (electrons $cm^{-3}$), $Y=f_h/f$ and $f_h$ is the gyrofrequency ($MHz$). Variable $\tau$ parameterises the trajectory, $\underline{x}$ is a position vector, $\underline{u}$ is a vector that is normal to the wavefront and $\underline{v}$ is a unit vector in the direction of the magnetic field. Quantities $J$, $K$, $L$ and $M$ are defined by

$$J = 2[2(1-X-Y^2)p + Y(1+(\underline{v}.\underline{u})^2)], \quad K = -2X(\underline{u}.\underline{v})(pY-1) \tag{3}$$

$$L = Y^2(1-Y^2)p^2 - 2(1-X-Y^2)p - Y, \quad M = 2Xp(pY-1) \tag{4}$$

where $p$ satisfies the quadratic

$$(1-X-Y^2)p^2 + Y(1+(\underline{v}.\underline{u})^2)p - (\underline{v}.\underline{u})^2 = 0 \tag{5}$$

and the different roots correspond to the ordinary and extraordinary rays.

In the current work, the Earth's magnetic field is represented by a tilted dipole

$$\underline{B} = -\nabla\left(\frac{\underline{M}.(\underline{x}-\underline{x}_0)}{|\underline{x}-\underline{x}_0|^3}\right) \tag{6}$$

where $\underline{M}$ is the magnetic moment (Davies, 1990) and $\underline{x}_0$ is the position of the dipole.
The phase path $P$ satisfies

$$\frac{dP}{d\tau} = \mu \cos\alpha \frac{ds}{d\tau} \tag{7}$$

the group path $P'$

$$\frac{dP'}{d\tau} = \mu' \cos\alpha \frac{ds}{d\tau} \tag{8}$$

and the ionospheric doppler shift $\Delta f$ (Bennett, 1967)

$$\frac{d\Delta f}{d\tau} = -\frac{f}{c}\frac{\partial \mu}{\partial t} \cos\alpha \frac{ds}{d\tau} \tag{9}$$

where $s$ is the distance along the ray ($ds = |d\underline{x}|$) and $\alpha$ is the angle between the wave normal and the ray direction. The refractive index $\mu$ is derived from the Appleton-Hartree equation (Davies, 1990)

$$\mu^2 = 1 - \frac{2X(1-X)}{2(1-X) - Y^2\sin^2\beta \pm \sqrt{Y^4\sin^4\beta + 4Y^2(1-X)^2\cos^2\beta}} \tag{10}$$

where $\beta$ is the angle between the wave normal and the magnetic field of Earth. The expression for the group refractive index $\mu'$ is derived from $\mu' = \frac{d}{df}(f\mu)$.

In the case that the magnetic field can be ignored, the trajectories are calculated according to

$$\frac{dx_i}{d\tau} = u_i \tag{11}$$

and

$$\frac{du_i}{d\tau} = -\frac{1}{2}\frac{\partial X}{\partial x_i} \tag{12}$$

where the new $\tau$ can be interpreted as the group path.

# 3 IMPLEMENTATION

The two most important issues in the implementation of the above scheme are the choice of ionospheric representation and the technique for solving the ODE's. In the present work, the ODE's are solved by a Runge-Kutta-Fehlberg (RKF) scheme that can be described as follows. Consider the vector system

$$\frac{d\underline{z}}{dt} = \underline{f}(\underline{z}, t) \tag{13}$$

with $\underline{z}=\underline{z}_k$ at time $t_k$. Form, in order, the vectors

$$\underline{k}_1 = h\underline{f}\left(t_k\ ,\ \underline{z}_k\right) \tag{14}$$

$$\underline{k}_2 = h\underline{f}\left(t_k+\frac{1}{4}h\ ,\ \underline{z}_k+\frac{1}{4}\underline{k}_1\right) \tag{15}$$

$$\underline{k}_3 = h\underline{f}\left(t_k+\frac{3}{8}h\ ,\ \underline{z}_k+\frac{3}{32}\underline{k}_1+\frac{9}{32}\underline{k}_2\right) \tag{16}$$

$$\underline{k}_4 = h\underline{f}\left(t_k+\frac{12}{13}h\ ,\ \underline{z}_k+\frac{1932}{2197}\underline{k}_1-\frac{7200}{2197}\underline{k}_2+\frac{7296}{2197}\underline{k}_3\right) \tag{17}$$

$$\underline{k}_5 = h\underline{f}\left(t_k+h\ ,\ \underline{z}_k+\frac{439}{216}\underline{k}_1-8\underline{k}_2+\frac{3680}{513}\underline{k}_3-\frac{845}{4104}\underline{k}_4\right) \tag{18}$$

$$\underline{k}_6 = h\underline{f}\left(t_k+\frac{1}{2}h\ ,\ \underline{z}_k-\frac{8}{27}\underline{k}_1+2\underline{k}_2-\frac{3544}{2565}\underline{k}_3+\frac{1859}{4104}\underline{k}_4-\frac{11}{40}\underline{k}_5\right) \tag{19}$$

These will yield fourth order

$$\underline{z}'_{k+1} = \underline{z}_k + \frac{25}{216}\underline{k}_1 + \frac{1048}{2565}\underline{k}_3 + \frac{2197}{4104}\underline{k}_4 - \frac{1}{5}\underline{k}_5 \tag{20}$$

and fifth order

$$\underline{z}''_{k+1} = \underline{z}_k + \frac{16}{135}\underline{k}_1 + \frac{6656}{12825}\underline{k}_3 + \frac{28561}{56430}\underline{k}_4 - \frac{9}{50}\underline{k}_5 + \frac{2}{55}\underline{k}_6 \tag{21}$$

Runge-Kutta approximations to the solution $\underline{z}=\underline{z}_{k+1}$ at time $t_{k+1}=t_k+h$. An estimate of the error incurred in the fourth order scheme is given by the difference between the fourth and fifth order approximations. In the RKF method, the steplength is adjusted such that this error remains below a preassigned value. The fifth order solution is used, however, for improved accuracy. The resulting scheme is extremely flexible and can adjust to the demands of a highly varying ionosphere. In the present application, the vector $\underline{z}$ has nine components. These consist of the position and wave normal components together with the phase path, the group path and the Doppler shift. The corresponding ODE's are given by equations (1), (2), (7), (8) and (9).

Three options for the representation of ionospheric electron density have been incorporated into the current computer procedure. The simplest of these allows the electron density to be defined through the function subroutine ELDEN. This is an extremely general facility that can accept most $C^1$ functions. (Since numerical derivatives are used in this option, it might experience difficulties if the ionosphere has extremely rapid variations.)

The second option provides for a description in terms of layer parameters. These are defined on a regular geographic grid (fixed increments in longitude and latitude) for two time slices. There are three layers (E, F1 and F2) which are defined through their heights (hmE, hmF1 and hmF2), the plasma frequencies at these heights (foE, foF1 and foF2) and the layer thicknesses (ymE, ymF1 and ymF2). The electron density profile is composed of Chapman layers, that is

$$N = N_1 Ch\left(\frac{1}{2}, \frac{2(h-hmE)}{ymE}\right) + N_2 Ch\left(\frac{1}{2}, \frac{2(h-hmF1)}{ymF1}\right) + N_3 Ch\left(1, \frac{\sqrt{2}(h-hmF2)}{ymF2}\right) \quad (22)$$

where

$$Ch(C,x) = \exp(C(1-x-\exp(-x))) \quad (23)$$

and the coefficients $N_i$ are chosen such the electron density has the correct values at the layer heights (a plasma frequency of $f_c$ $MHz$ corresponds to electron density of $12407 f_c^2$ electrons per $cm^3$). In order to estimate the density values between the grid points, some sort of interpolation scheme is required. Although sophisticated approaches such as cubic splines are desirable, these are too inefficient for a general purpose procedure. Instead, the multi-dimensional interpolation is built out of a series of localised one dimensional cubic interpolations. Consider the meridian that passes through the point of interest and calculate its intersection with the four closest parallels that pass through grid points. Derive values at the points of intersection by use of one dimensional interpolations along each of the parallels. The value at the point of interest is obtained by means of a one dimensional interpolation between the points of intersection.

There are several possibilities for the one dimensional interpolation and two options are included in the current procedure. The first provides an approximation with $C^1$ continuity. Let the grid points be located a unit distance apart and the desired point lie between the middle points a distance $u$ from the second point. If the data has values $e_1$, $e_2$, $e_3$ and $e_4$ at the interpolation points, the value at the desired point is approximately

$$e_0\left(u^2 - \frac{u}{2} - \frac{u^3}{2}\right) + e_1\left(1 - \frac{5u^2}{2} + \frac{3u^3}{2}\right) + e_3\left(\frac{u}{2} + 2u^2 - \frac{3u^3}{2}\right) + e_2\left(\frac{u^3}{2} - \frac{u^2}{2}\right) \quad (24)$$

If the data points are sparse, it has been found advisable to sacrifice the $C^1$ continuity and to use Lagrange interpolation instead. In this case, the value at the desired point is approximately

$$-\frac{e_1}{6}u(u-1)(u-2) + \frac{e_2}{2}(u+1)(u-1)(u-2) - \frac{e_3}{2}(u+1)u(u-2) + \frac{e_4}{6}(u+1)u(u-1) \quad (25)$$

The third option for ionospheric representation replaces the Chapman layers with a set of samples above each geographical grid point. Sampling is uniform and so the above geographic interpolation scheme can be extended to include the extra dimension. This option is by far the most efficient and is recommended when a large number of rays are to be traced.

# 4 THE PROCEDURE

The ray tracing procedure has been implemented as the FORTRAN subroutine HASEL. Each call to this subroutine will trace rays that are launched at a given elevation, in a given direction, from a given starting point and with a given wave frequency. The rays can be traced for up to three hops when the effect of Earth's magnetic field is included and for up to ten hops when not. A typical call to the subroutine is of the form

```
CALL HASEL(Long,Lat,Elav,Bear,Freq,Cha,Nos,Out,Type,Tol,M,L)
```

where the input parameters are defined as follows:

1) Long is a real*8 variable that specifies the geographic longitude of the start point (in degrees).
2) Lat is a real*8 variable that specifies the geographic latitude of the start point (in degrees).
3) Elav is a real*8 variable that specifies the initial elevation of the ray (in degrees).
4) Bear is a real*8 variable that specifies the initial bearing of the ray (degrees East from North).
5) Freq is a real*8 variable that specifies the wave frequency (in MHz).
6) Tol is a real*8 variable that specifies the tolerance for the RKF method at each step (in kilometres).
7) Cha is a character variable that has the value 'y' if the effect of Earth's magnetic field is to be included and 'n' if not.
8) Nos is an integer variable that specifies the number of hops that are required (this parameter is altered on exit from the subroutine).
9) M is an integer variable that specifies the type of ionospheric representation. The analytic representation using subroutine ELDEN is chosen when M=7, the layer model defined on a geographic grid when M=6 or -6 and the representation in terms of height samples on a geographic grid when M=10 or -10. (Negative values of M invoke the Lagrange interpolation option.)
10) L is an integer variable that specifies the channel number on which details concerning the ray trajectories are to be output (if L < 1 this information will be omitted). For each step of the RKF scheme, this option outputs the height above the ground, the longitude and the latitude. In addition, the angle between the ray and Earth's magnetic field is output.

The function option (M=7) defines the ionosphere through the real*8 function subroutine ELDEN. This subroutine has four real*8 arguments consisting of time elapsed (measured in seconds from the start of ray tracing), longitude (in degrees), latitude (in degrees) and distance from Earth's centre (in kilometres). A typical example of ELDEN is as follows:

```
REAL*8 ELDEN(t,long,lat,r)
implicit real*8 (a-l,n-z)
common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
H=100.0d0
hmF2=350.0d0
foF2=12.0d0
vel=.04
x=(r-6378.1d0-hmF2-vel*t)/H
NF2=12407*foF2**2
ELDEN=NF2*exp(1.0d0-x-exp(-x))
return
end
```

This represents a Chapman layer ionosphere with peak height that rises at a constant speed.

The common block is optional, but can be used to pass back information concerning the layer heights and hence allow a meaningful labelling of rays.

The grid options (M = 6,-6,10 or -10) require further information to be passed from the calling program through a common block. The following statement will need to be placed in the calling program

```
COMMON /grid/MinLa,MinLo,MinR,DLa,DLo,DR,DT,Grid,NR,NLa,NLo
```

with the parameters defined as follows:

a) MinLa is a real*8 variable that specifies the minimum latitude on the grid (in degrees).

b) MinLo is a real*8 variable that specifies the minimum longitude on the grid (in degrees).

c) MinR is a real*8 variable that specifies the minimum distance (in kilometres) of the grid from Earth's centre (options 10 or -10).

d) DLa is a real*8 variable that specifies the increment in latitude on the grid (in degrees).

e) DLo is a real*8 variable that specifies the increment in longitude on the grid (in degrees).

f) DR is a real*8 variable that specifies the increment in height on the grid (in kilometres).

g) DT is a real*8 variable that specifies the increment in time between the time slices (in seconds).

h) GRID is a three argument real*4 array that contains grid values of either the layer parameters or the electron density. The third argument of the array fixes the time slice and can only have the values 1 or 2. Geographical location is specified by the second argument. As this argument increases, it runs eastwards along rows of constant latitude in turn. The rows start at the minimum latitude and move upwards. The meaning of the first argument depends on the value of M. For M=6 or -6, the argument values 1 to 9 label the layer parameters foE, hmE, ymE, foF1, hmF1, ymF1, foF2, hmF2, ymF2 in order (frequencies in MHz and lengths in kilometres). In the case that M=10 or -10, the first argument labels the height samples of electron density (electrons $cm^{-3}$) starting at the lowest height and moving upwards. The dimensions of the first and second arguments should be set to the largest values that will be required. (For options M=6,-6,10 or -10, the array GRID must be defined in the calling program.)

i) NR is an integer variable that specifies the number sample heights.

j) NLa is an integer variable that specifies the number of sample latitudes.

k) NLo is an integer variable that specifies the number of sample longitudes.

The output from HASEL is contained in the variables Out, Type and Nos. These are defined as follows:

1) Nos is an integer variable that specifies the total number of rays traced.

2) Out is a two argument real*8 array (dimensions 11 and 14 respectively) that contains information about the rays. Each ray is labelled by the second argument (values from 1 to Nos). The first argument labels the ray attributes as follows:

Out(1,*) = Doppler shift (Hz)
Out(2,*) = ground range (kilometres)
Out(3,*) = group path (kilometres)
Out(4,*) = phase path (kilometres)
Out(5,*) = maximum height (kilometres)

Out(6,*) = initial elevation (degrees)
Out(7,*) = final elevation (degrees)
Out(8,*) = final longitude (degrees)
Out(9,*) = final latitude (degrees)
Out(10,*) = bearing (degrees East from North)
Out(11,*) = wave frequency (MHz).

3) Type is a character*21 array (dimension 14) that contains information concerning the mode of the ray that corresponds to the argument.

There is an additional facility for situations where the ionosphere is periodic in longitude. If the value of variable nos is negated, the geographic mesh (options m= 10, -10, 6 or -6 only) will be treated as one period of a periodic structure. This facility is extremely useful for an ionosphere that does not vary with longitude (set nlo=1 in this case).

## 5 CONCLUSIONS

The present report has described a ray tracing procedure that can analyse the propagation of radiowaves through a variety of ionospheres and includes the effect of the Earth's magnetic field. There are numerous uses for the procedure, mainly in the areas of propagation prediction and ionospheric research. In particular, two recent applications are the simulation of backscatter ionograms and some investigations of the Doppler characteristics of radiowaves that propagate through travelling ionospheric disturbances. A version that estimates irregularity induced Doppler spreading has also been developed and this has been used to investigate the phenomenon of Doppler spread clutter.

THIS IS A BLANK PAGE

## REFERENCES


1. Haselgrove, J — (1955) 'Ray theory and a new method for ray tracing', in Physics of the ionosphere, pp 355-64, Physical Society, London.

2. Haselgrove, C.B and Haselgrove, J — 'Twisted Ray Paths in the Ionosphere', Proc.Phys.Soc. (London), Vol.75, pp 357-361, 1960.

3. Haselgrove, J — 'The Hamilton Ray Path Equations', in J.Atmos.Terr.Phys, Vol.25, pp 397-399, 1963.

4. Davies, K — (1990) 'Ionospheric Radio Waves', IEE Electromagnetic Waves Series 31, Peter Peregrinus, London.

5. Bennett, J.A — 'The calculation of Doppler Shifts due to a changing ionosphere', J.Atmos.Terr.Phys., Vol.29, pp 887-891, 1967.

THIS IS A BLANK PAGE

# APPENDIX A The Ray Tracing Software

```
************************************************************************
***** HASEL (version 2.2) ********************************************
************************************************************************
* The following routine performs ionosheric ray tracing.. It can accept *
* a variety of ionospheres and uses a Runge-Kutta-Fehlberg scheme to   *
* solve the Haselgrove equations.                                      *
************************************************************************
* Unless otherwise stated, all distances are in kilometres, angles in  *
* degrees, frequencies in MegaHertz and time in seconds.               *
************************************************************************
***** input **********************************************************
* REAL*8                                                               *
*  along = longitude of the start point (degrees E of W)               *
*  alat  = latitude of start point (degrees, -ve for S of Equator)     *
*  elav  = initial elevation (degrees from horizontal)                 *
*  bear  = initial bearing (degrees east from north)                   *
*  f     = wave frequency                                              *
*  tol   = tolerance (Kms) at each step of raytracing (a value of      *
*          1.d-6 is sufficient in most cases)                          *
* CHARACTER                                                            *
*  cha   = 'y' if magnetic fields are to be included, 'n' otherwise    *
* INTEGER                                                              *
*  nos   = number of hops (changed on output to total number of rays)  *
*  l     = channel number for ray trajectory information (0 if no      *
*          information required)                                       *
*  m     =  7 for an analytic model that is described by REAL*8        *
*           function ELDEN                                             *
*           6 for a description in terms    :                         *
*           of layer parameters on a 2D     : information passed      *
*           grid at two time slices         : via the common          *
*          10 for a description in terms    : block GRID              *
*           of electron density values      :                         *
*           on a 3D grid at two time        :                         *
*           slices                          :                         *
************************************************************************
***** output *********************************************************
* REAL*8                                                               *
*  out(1,*)  = doppler shift (Hz)  :                                   *
*  out(2,*)  = ground range        :                                   *
*  out(3,*)  = group path          :                                   *
*  out(4,*)  = phase path,         :                                   *
*  out(5,*)  = maximum height      :                                   *
*  out(6,*)  = initial elevation   : * refers to the ray number        *
*  out(7,*)  = final elevation     :                                   *
*  out(8,*)  = final longitude     :                                   *
*  out(9,*)  = final latitude      :                                   *
*  out(10,*) = bearing             :                                   *
*  out(11,*) = frequency           :                                   *
* CHARACTER*21                     :                                   *
*  typo(*)   = mode                :                                   *
* INTEGER                                                              *
*  nos       = number of rays traced                                   *
* Note:                                                                *
*     The program outputs the ray trajectory and values of the         *
*     angle between the ray and Earth's magnetic field. Each           *
*     line of output will consist of a height, a longitude, a          *
*     latitude and an angle.                                           *
************************************************************************
***** subroutines required *******************************************
*  LAYERHT, RKF, FUNC, LAYERX, TERP, DTERP, ELDEN, ELECTRONR, ELDENX   *
*  ELECTRON, MAGNETIC                                                  *
************************************************************************
***** common block ( to be set by user under certain options) **********
* The parameters of the common block GRID are, in order,               *
* REAL*8                                                               *
*  blhla = lowest value of latitude                                    *
*  blhlo = lowest value of longitude                                   *
*  rmg   =  height of ionosphere above the Earth centre                *
*  dla   =  latitude increment                                         *
*  dlo   =  longitude increment                                        *
```

```
*  drg    =  height increment (for m = 10 only)                          *
*  dtz    =  time between temporal slices                                *
* INTEGER                                                                *
*  nr     = number of height samples for m = 10 or 9 for m = 6           *
*  nla    = number of latitude samples                                   *
*  nlo    = number of longitude samples                                  *
* REAL*4                                                                 *
*  yp(i1,i2,i3) =     electron density samples if m = 10                 *
*                     or layer parameter samples if m = 6                *
*                                                                        *
* for m=6   yp contains the i1'th layer parameter in the list            *
*           foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2 (frequencies       *
*           in MHz and heights in Km)                                    *
*                                                                        *
* for m=10  yp contains electron density (electrons per cubic cm)        *
*           samples at height rmg+(i1-1)*drg                             *
*                                                                        *
* i2 = (i-1)*nlo+j  for the samples at latitude blhla+(i-1)*dla          *
* and longitude blhlo+(j-1)*dlo                                          *
*                                                                        *
* i3 labels the time slice (1 or 2)                                      *
*                                                                        *
* Notes:                                                                 *
* 1) Maximum values of i1 and i2 have been set at 61 and 961. If         *
*    these limits are to be changed, remember to alter for all           *
*    occurences of yp.                                                   *
* 2) A maximum value of 10Km is advised for drg , 5 degrees for dla      *
*    and 5 degrees for dlo.                                              *
**************************************************************************
***** switches ********************************************************
*  1) If tol = -tolerence the program will execute faster, but the       *
*     doppler will be based on a stationary ionosphere.                  *
*  2) If l<1 no ray path data will be output.                            *
*  3) If m<0 Lagrange interpolation is used.                             *
*  4) If nos<0 the ionosphere will be given a periodic extension in      *
*     longitude.                                                         *
**************************************************************************
***** common blocks that are internal **********************************
* DAT holds (in order) current values (real*8) of layer parameters       *
* foE, hmE, ymE, foF1, hmF1, ymF1, foF2, hmF2 and ymF2. The final two    *
* parameters are the Earths radius (real*8 in Kms) and the current       *
* model (an integer).                                                    *
* APPROX consists of a single character variable that has value 'L'      *
* for Lagrange interpolation and 'S' for smooth jointed interpolation.   *
* MAG has eight real*8 variables followed by an integer variable. The    *
* real variables are the current gyrofrequency (MHz), the wave           *
* frequency (MHz), the Cartesian coordinates (Earth centred) of the      *
* magnetic dipole and the dipole moments. The integer variable has       *
* the value 0 when the magnetic is to be ignored, a value 1 when         *
* ordinary rays are to be traced and value -1 when extraordinary rays    *
* are to be traced.                                                      *
**************************************************************************
      subroutine HASEL(along,alat,elav,bear,f,cha,nos,out,typo,tol,m,l)
      implicit real*8 (a-h,o-z)
       real*4 yp(61,961,2)
      character type*21,typo*21,ch1,c11*21,c12*21,cha,app
       dimension x(11),ytmp(11),xx(11),xtmp(11),out(11,14),typo(14)
     &,v(3),ryx(3,3)
       common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
       common /grid/blhla,blhlo,rmg,dla,dlo,drg,dtz,yp,nr,nla,nlo
       common /approx/app
       common /mag/fhs,fr,xm,ym,zm,px,py,pz,ir
      re=6378.135
      scale=45./atan(1.)
      n=9
      noz=nos
      fr=f
       if(m.lt.0) then
      app='S'
       else
      app='L'
       endif
      model=abs(m)
      if(model.eq.7) dtz=1.
```

```
       if(tol.lt.0.0) then
       dtw=dtz
       dtz=0.0
       endif
      tolx=abs(tol)
***** dipole moment = (pxt,pyt,pzt) and dipole location = (xm,ym,zm) ***
      dlat=77.75
      dlong=295.75
      pxt=cos(dlat/scale)*cos(dlong/scale)
      pyt=cos(dlat/scale)*sin(dlong/scale)
      pzt=sin(dlat/scale)
      xm=-434.66
      ym=199.47
      zm=80.25
       fhs=2.8*.31*re**3
      px=fhs*pxt/fr
      py=fhs*pyt/fr
      pz=fhs*pzt/fr
      if(cha.ne.'y') fhs=0.0
*************************************************************************
      trhla=blhla+real(nla-1)*dla
      trhlo=blhlo+real(nlo-1)*dlo
      nhop=abs(nos)
       if(cha.eq.'n') then
       ira=0
       irb=0
       else
       ira=1
       irb=2
       if(abs(nos).eq.2) nhop=3
       if(abs(nos).ge.3) nhop=7
       endif
      nos=0
*************************************************************************
      do 909 irr=ira,irb
***** ray type loop *****************************************************
       if(irr.eq.0) then
       ch1='N'
       ir=0
       else if(irr.eq.1) then
       ch1='O'
       ir=1
       else if(irr.eq.2) then
       ch1='X'
       ir=-1
       endif
      dr=sin(elav/scale)
      dt=-cos(elav/scale)*cos(bear/scale)
      dp=cos(elav/scale)*sin(bear/scale)
      phi=atan(1.)*along/45.
      theta=2.*atan(1.)-atan(1.)*alat/45.
      x(1)=re*sin(theta)*cos(phi)
      x(2)=re*sin(theta)*sin(phi)
      x(3)=re*cos(theta)
      x1=x(1)
      x2=x(2)
      x3=x(3)
      x(4)=dr*sin(theta)*cos(phi)+dt*cos(theta)*cos(phi)-dp*sin(phi)
      x(5)=dr*sin(theta)*sin(phi)+dt*cos(theta)*sin(phi)+dp*cos(phi)
      x(6)=dr*cos(theta)-dt*sin(theta)
      t=0.0
      ita=2
      istep=1
      type='              '
      x(7)=0.0
      x(8)=0.0
      x(9)=0.0
*************************************************************************
      do 979 ihop=1,nhop
***** hop loop **********************************************************
      zmax=0.0
      sc=sqrt(x(4)**2+x(5)**2+x(6)**2)
      x(4)=x(4)/sc
      x(5)=x(5)/sc
```

```
      x(6)=x(6)/sc
       if(cha.eq.'y'.and.ihop.gt.1) then
        if(ihop.eq.2) then
          do k=1,n
           xtmp(k)=x(k)
          enddo
         ir=1
         ch1='O'
        else
         if(ihop.eq.3.or.ihop.eq.4.or.ihop.eq.6) ir=-1
         if(ihop.eq.5.or.ihop.eq.7) ir=1
          if(ihop.eq.3) then
            do k=1,n
             ytmp(k)=x(k)
             x(k)=xtmp(k)
            enddo
           cl1=type
           ita1=itb+1
          endif
          if(ihop.eq.4) then
            do k=1,n
             xtmp(k)=x(k)
            enddo
           cl2=type
           ita2=itb+1
          endif
           if(ir.eq.1) ch1='O'
           if(ir.eq.-1) ch1='X'
          if(ihop.eq.4.or.ihop.eq.5) then
            do k=1,n
             x(k)=ytmp(k)
            enddo
           type=cl1
           ita=ita1
          endif
          if(ihop.eq.6.or.ihop.eq.7) then
            do k=1,n
             x(k)=xtmp(k)
            enddo
           type=cl2
           ita=ita2
          endif
        endif
       endif
      zad=0.0
      zed=0.0
      itb=ita+1
      type(ita:itb+1)='S'//ch1//' '
      itx=2
       if(ch1.eq.'N') then
      type(itb:itb)=' '
      itb=itb-1
      itx=itx-1
       endif
***** start of calculations for an individual hop **********************
 999  istep=istep+1
       do i=1,6
      xx(i)=x(i)
       enddo
      pi=x(7)
      dopx=x(8)
      pp=x(9)
      zbd=zad
      zad=zed
      xlong=rlong
      xlat=rlat
***** ODE solver ********************************************************
      hmax=50.
      hmin=.01
      h=hmax
       call rkf(t,x,n,h,tolx,hmin,hmax)
* x(1) to x(3) = Cartesian coordinates of current ray position
* x(7) to x(9) = group path, doppler shift and phase path at this
* location
```

```
*************************************************************************
      red=sqrt(x(1)**2+x(2)**2+x(3)**2)
      zed1=zed
      zed=red-re
      rlat=90.-acos(x(3)/red)*scale
       if(abs(x(1)).lt.1.0d-10) then
      rlong=atan2(x(2),1.0d-10)*scale
       else
      rlong=atan2(x(2),x(1))*scale
       endif
      if(rlong.lt.0.0) rlong=rlong+360.0
       if(1.gt.0) then
        call magnetic(x,v,ryx,ha,dip.dcl,n)
        vu=x(4)*v(1)+x(5)*v(2)+x(6)*v(3)
        xt=sqrt(abs(x(4)**2+x(5)**2+x(6)**2))
        angle=acos(.9999999*vu/(ha*xt))*scale
        if(zed.gt.0.0) write(1,fmt=' 4f15.7)') zed,rlong,rlat,angle
       endif
      if((rlat.gt.trhla.or.rlat.lt.blhla).and.model.ne.7) go to 909
      if((rlong.gt.trhlo.or.rlong.lt.blhlo)
     &.and.model.ne.7.and.noz.gt.0) go to 909
*************************************************************************
***** At the current stage of solution
***** zed= altitude of the ray
***** rlong = longitude of ray
***** rlat = latitude of ray
*************************************************************************
***** label for highest reflection layer of current hop ***************
      if(zad.gt.max(zbd,zed,zmax)) then
       if(model.eq.10) then
         call layerparm(xlong,xlat,h1,h2,h3,f1,f2,f3)
         hmE=h1
         hmF1=h2
         hmF2=h3
       endif
       if(zad.le.hmE) then
        itb=ita+1
        type(ita:itb+1)='E'//ch1//' '
        itx=2
       else if(zad.le.hmF1) then
        itb=ita+2
        type(ita:itb)='F1'//ch1
        itx=3
       else if(zad.le.hmF2) then
        itb=ita+2
        type(ita:itb)='F2'//ch1
        itx=3
       else
        itb=ita+1
        type(ita:itb+1)='S'//ch1//' '
        itx=2
       endif
       if(ch1.eq.'N') then
        type(itb:itb)=' '
        itb=itb-1
        itx=itx-1
       endif
      endif
*************************************************************************
      zmax=max(zad,zmax)
      if(zed.ge.600.) go to 909
      if(zed.gt.0.) go to 999
      if(ihop.eq.1.or.cha.ne.'y') ita=ita+itx
***** interpolation for values at ground level ************************
      pop=zed1/(zed1-zed)
       do i=1,6
      x(i)=xx(i)+pop*(x(i)-xx(i))
       enddo
      x1=0.0
      xx1=0.0
      cosa=0.0
       do i=1,3
      x1=x1+x(i)**2
      xx1=xx1+x(i+3)**2
```

```
       cosa=cosa+x(i)*x(i+3)
       enddo
      elavf=abs(90.-45.*acos(cosa/sqrt(xl*xxl))/atan(1.))
      sd=sqrt((x(1)-x1)**2+(x(2)-x2)**2+(x(3)-x3)**2)
      sd=2.*re*asin(.5*sd/re)
      pl=pl+pop*(x(7)-pl)
      pp=pp+pop*(x(9)-pp)
      dopx=dopx+pop*(x(8)-dopx)
      x(7)=pl
      x(8)=dopx
      x(9)=pp
      dopx=1.0d6*dopx
      xlong=xlong+pop*(rlong-xlong)
      xlat=xlat+pop*(rlat-xlat)
      nos=nos+1
      out(1,nos)=dopx
      out(2,nos)=sd
      out(3,nos)=pl
      out(4,nos)=pp
      out(5,nos)=zmax
      out(6,nos)=elav
      out(7,nos)=elavf
      out(8,nos)=xlong
      out(9,nos)=xlat
      out(10,nos)=bear
      out(11,nos)=fr
      typo(nos)=type
      ss=x(1)**2+x(2)**2+x(3)**2
      xr=x(4)*x(1)+x(5)*x(2)+x(6)*x(3)
      x(4)=x(4)-2.*xr*x(1)/ss
      x(5)=x(5)-2.*xr*x(2)/ss
      x(6)=x(6)-2.*xr*x(3)/ss
      if(1.gt.0) then
      zod=0.0
      write(1,fmt='(4f15.7)') zod,xlong,xlat,angle
      endif
 979  continue
 909  continue
      if(tol.lt.0.0) dtz=dtw
      return
      end

***** LAYERPARM ***************************************************************
*******************************************************************************
***** layer parameters at a given location ************************************
*******************************************************************************
***** input *******************************************************************
* real*8                                                                       *
* elong = longitude (deg) of sample point                                      *
* elat = latitude (deg) of sample point                                        *
***** output ******************************************************************
* real*8                                                                       *
* h1 = height of E layer                                                       *
* h2 = height of F1 layer                                                      *
* h3 = height of F2 layer                                                      *
* f1 = critical frequency of E layer                                           *
* f2 = critical frequency of F1 layer                                          *
* f3 = critical frequency of F2 layer                                          *
*******************************************************************************
***** subroutines required ****************************************************
* ELDENX                                                                       *
*******************************************************************************
***** common block ************************************************************
* GRID (defined in HASEL)                                                      *
*******************************************************************************
      subroutine layerparm(elong,elat,h1,h2,h3,f1,f2,f3)
      implicit real*8 (a-h,o-z)
        real*4 yp(61,961,2)
       common /grid/blhla,blhlo,rmg,dla,dlo,drg,dtz,yp,nr,nla,nlo
      re=6378.135
      r0=rmg
       e0=eldenx(elong,elat,r0,d10,dthet,dphi,1)
       r1=r0+drg
       e1=eldenx(elong,elat,r1,d11,dthet,dphi 1)
```

```
      h0=rmg-re
      h1=0.0
      h2=0.0
      h3=0.0
       ea=0.0
       eb=0.0
      ec=0.0
        do i=3,nr
       r2=rmg+drg*(i-1)
       e2=eldenx(elong,elat,r2,d12,dthet,dphi,1)
       d21=(d12-d10)/(r2-r0)
       hh=0.0
       ee=0.0
       hz=0.0
        if(d11*d10.le.1.d-33.and.abs(d11-d10).gt.1.d-33) then
***** check for maximum ***************************************************
          hh=(d11*r0-d10*r1)/(d11-d10)-re
          ee=(d11*e0-d10*e1)/(d11-d10)
           if(hh.gt.100.0) then
             if(hh.lt.140.) then
               h1=hh
               ea=ee
             else
                if(hh.lt.230.) then
               h2=hh
               eb=ee
                else
               h3=hh
               ec=ee
                endif
             endif
           endif
         else
***** check for point of inflection ***********************************
           if(i.gt.3) then
             if(r1-re.lt.230.) then
               if(d21*d20.le.1.d-33.and.abs(d21-d20).gt.1.d-33) then
              hz=(d21*r0-d20*r1)/(d21-d20)-re
              ez=(d21*e0-d20*e1)/(d21-d20)
               if(hz.gt.100.0.and.hz.lt.230.) then
                 if(hz.ge.140.) then
                   if(h2.lt.1.) then
                            h2=hz
                            eb=ez
                   endif
                 else
                   if(h1.lt.1.) then
                            h1=hz
                            ea=ez
                   endif
                 endif
               endif
               endif
             endif
           endif
         endif
       d10=d11
       d11=d12
       d20=d21
       e0=e1
       e1=e2
       r0=r1
       r1=r2
       enddo
       if(h1.lt.h0) h1=h0
      if(h2.lt.h0) h2=h1
      if(h3.lt.h0) h3=h2
       if(ec.lt.1.) then
       ec=e2
       h3=r2-re
       endif
       f1=sqrt(abs(ea*80.6d-6))
       f2=sqrt(abs(eb*80.6d-6))
       f3=sqrt(abs(ec*80.6d-6))
```

```
      return
      end

***** ELECTRON ********************************************************************
************************************************************************************
***** electron density and its gradient ********************************************
************************************************************************************
***** input ************************************************************************
* real*8                                                                            *
* x(*) = Cartesian coordinates (Earth centred) of sample point                      *
***** output ***********************************************************************
* real*8                                                                            *
* en = electron density (electrons per cubic cm)                                    *
* dnx(*) = Cartesian derivatives of electron density                                *
************************************************************************************
***** subroutines required *********************************************************
* ELDENX, ELDEN, LAYERX and ELECTRONR                                               *
************************************************************************************
***** common blocks ****************************************************************
* GRID and DAT (defined in HASEL)                                                   *
************************************************************************************
      subroutine electron(x,en,dnx)
      implicit real*8 (a-h,o-z)
      dimension x(3),dnx(4)
       real*4 yp(61,961,2)
       common /grid/blhla,blhlo,rmg,dla,dlo,drg,dtz,yp,nr,nla,nlo
       common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
      r=sqrt(x(1)**2+x(2)**2+x(3)**2)
      scale=45./atan(1.)
      theta=acos(x(3)/r)
      phi=atan2(x(2),x(1))
      elat=90.-scale*theta
      elong=scale*phi
      if(elong.lt.0.0) elong=elong+360.
***** Parameter da should be less than 20% of the smallest geographic **
***** scale (in degrees).                                              **
      da=.002
       if(model.eq.10) then
       en=eldenx(elong,elat,r,diffr,dthet,dphi,1)
       dthet=scale*dthet
       dphi=scale*dphi
        if(dtz.gt.1.d-20) then
         enp=eldenx(elong,elat,r,d1,d2,d3,-2)
         enm=en
        endif
       else
        if(model.eq.7) then
***** Parameter dr should be less than 20% of the smallest height ******
***** scale (in kilometers).                                      ******
         dr=.2
         t=0.0
         enpp=elden(t,elong+da,elat,r)
         enpm=elden(t,elong-da,elat,r)
         entp=elden(t,elong,elat+da,r)
         entm=elden(t,elong,elat-da,r)
          if(dtz.gt.1.d-20) then
           enp=elden(t+.5d0*dtz,elong,elat,r)
           enm=elden(t-.5d0*dtz,elong,elat,r)
          endif
        diffr=.5*(elden(t,elong,elat,r+dr)-elden(t,elong,elat,r-dr))/dr
        else
         call layerx(elong+da,elat,1)
         call electronr(r,enpp,dpp)
         call layerx(elong-da,elat,1)
         call electronr(r,enpm,dpm)
         call layerx(elong,elat+da,1)
         call electronr(r,entp,dtp)
         call layerx(elong,elat-da,1)
         call electronr(r,entm,dtm)
          if(dtz.gt.1.d-20) then
           call layerx(elong,elat,-2)
           call electronr(r,enp,dq)
           enm=.25*(enpp+enpm+entp+entm)
          endif
```

```
        diffr=.25*(dpp+dpm+dtp+dtm)
       endif
      en=.25*(enpp+enpm+entp+entm)
      dphi=.5*scale*(enpp-enpm)/da
      dthet=-.5*scale*(entp-entm)/da
       endif
       if(dtz.lt.1.d-20) then
      dtime=dphi/(240.*scale)
       else
      dtime=(enp-enm)/dtz
       endif
     dnx(1)=(diffr*x(1)+dthet*cos(theta)*cos(phi)
    &-dphi*sin(phi)/sin(theta))/r
     dnx(2)=(diffr*x(2)+dthet*cos(theta)*sin(phi)
    &+dphi*cos(phi)/sin(theta))/r
     dnx(3)=(diffr*x(3)-dthet*sin(theta))/r
     dnx(4)=dtime
     return
     end

***** LAYERX ********************************************************************
*******************************************************************************
***** layer parameters from grid values ******************************************
*******************************************************************************
***** input **********************************************************************
* real*8                                                                       *
* elong = longitude (deg) of sample point                                     *
* elat = latitude (deg) of sample point                                       *
* integer                                                                      *
* iuu = time slice label (1 or 2)                                             *
*******************************************************************************
***** subroutine required ********************************************************
* TERP                                                                         *
*******************************************************************************
***** common blocks **************************************************************
* GRID and DAT (defined in HASEL)                                              *
*******************************************************************************
***** note ***********************************************************************
* The output is placed in the common block DAT                                 *
*******************************************************************************
      subroutine layerx(elong,elat,iuu)
      implicit real*8 (a-h,o-z)
       real*4 yp(61,961,2)
      dimension yz(4),yy(9)
       common /grid/blhla,blhlo,rmg,dla,dlo,drg,dtz,yp,nr,nla,nlo
       common /dat/yy,re,model
      lip=abs(iuu)
      dlong=elong-blhlo
      if(dlong.gt.360.) dlong=dlong-360.
      la=min(max(2,int((elat-blhla)/dla)+1),nla-2)
      lo=int(dlong/dlo)+1
      lo=lo-int((lo-1)/nlo)*nlo
      if(lo.lt.1) lo=lo+nlo
      dax=(elat-blhla)/dla-real(la-1)
      dox=dlong/dlo-real(lo-1)
      n0=(la-2)*nlo
      n1=n0+nlo
      n2=n1+nlo
      n3=n2+nlo
       do i=1,9
        do k=0,3
         kk=lo-1+k
         if(kk.lt.1) kk=kk+nlo
         if(kk.gt.nlo) kk=kk-nlo
         x0=yp(i,n0+kk,lip)
         x1=yp(i,n1+kk,lip)
         x2=yp(i,n2+kk,lip)
         x3=yp(i,n3+kk,lip)
         yz(k+1)=terp(dax,x0,x1,x2,x3)
        enddo
       yy(i)=terp(dox,yz(1),yz(2),yz(3),yz(4))
       enddo
      return
      end
```

```
***** ELDENX ********************************************************************
**********************************************************************************
***** electron density and gradients from grid values ******************************
**********************************************************************************
***** input **********************************************************************
* real*8                                                                         *
* elong = longitude (deg) of sample point                                        *
* elat = latitude (deg) of sample point                                          *
* r = distance from Earth centre (Kms)                                           *
* integer                                                                        *
* iuu = time slice label (1 or 2)                                                *
**********************************************************************************
***** output *********************************************************************
* real*8                                                                         *
* eldenx = electron density (electrons per cubic cm)                             *
* diffr = derivative in vertical direction                                       *
* dthet = derivative in counter latitudinal direction                            *
* dphi = derivative in longitudinal direction                                    *
**********************************************************************************
***** subroutines required *******************************************************
* TERP, DTERP                                                                    *
**********************************************************************************
***** common block ***************************************************************
* GRID (defined in HASEL)                                                        *
**********************************************************************************
      real*8 function eldenx(elong,elat,r,diffr,dthet,dphi,iuu)
      implicit real*8 (a-h,o-z)
       real*4 yp(61,961,2)
      dimension yz(4),yzt(4),e(4),dt(4),dp(4)
       common /grid/blhla,blhlo,rmg,dla,dlo,drg,dtz,yp,nr,nla,nlo
      lip=abs(iuu)
       if(r.lt.rmg.or.r.gt.rmg+(nr-1)*drg) then
       eldenx=0.0
       diffr=0.0
       dthet=0.0
       dphi=0.0
       else
       dlong=elong-blhlo
       if(dlong.gt.360.) dlong=dlong-360.
       la=min(max(2,int((elat-blhla)/dla)+1),nla-2)
       lo=int(dlong/dlo)+1
       lo=lo-int((lo-1)/nlo)*nlo
       if(lo.lt.1) lo=lo+nlo
       lr=min(max(1,int((r-rmg)/drg)+1),nr-2)
       dax=(elat-blhla)/dla-real(la-1)
       dox=dlong/dlo-real(lo-1)
       drx=(r-rmg)/drg-real(lr-1)
       n0=(la-2)*nlo
       n1=n0+nlo
       n2=n1+nlo
       n3=n2+nlo
        if(lr.lt.2) then
       nstart=0
        else
       nstart=-1
        endif
        do i=nstart,2
          do k=0,3
           kk=lo-1+k
           if(kk.lt.1) kk=kk+nlo
           if(kk.gt.nlo) kk=kk-nlo
           x0=yp(lr+i,n0+kk,lip)
           x1=yp(lr+i,n1+kk,lip)
           x2=yp(lr+i,n2+kk,lip)
           x3=yp(lr+i,n3+kk,lip)
           if(iuu.gt.0) yzt(k+1)=dterp(dax,x0,x1,x2,x3)
           yz(k+1)=terp(dax,x0,x1,x2,x3)
          enddo
         yy=terp(dox,yz(1),yz(2),yz(3),yz(4))
          if(iuu.gt.0) then
           yyt=terp(dox,yzt(1),yzt(2),yzt(3),yzt(4))
           yyp=dterp(dox,yz(1),yz(2),yz(3),yz(4))
          else
           yyt=0.
```

```
          yyp=0.
         endif
        e(i+2)=yy
        dt(i+2)=yyt
        dp(i+2)=yyp
       enddo
       if(lr.lt.2) then
        e(1)=0.0
        dt(1)=0.0
        dp(1)=0.0
       endif
      eldenx=terp(drx,e(1),e(2),e(3),e(4))
      if(eldenx.lt.0.0) then
       eldenx=0.0
       diffr=0.0
       dthet=0.0
       dphi=0.0
      else
        if(iuu.gt.0) then
         diffr=dterp(drx,e(1),e(2),e(3),e(4))
         dthet=-terp(drx,dt(1),dt(2),dt(3),dt(4))
         dphi=terp(drx,dp(1),dp(2),dp(3),dp(4))
        else
         diffr=0.0
         dthet=0.0
         dphi=0.0
        endif
      endif
       endif
     dthet=dthet/dla
     dphi=dphi/dlo
     diffr=diffr/drg
     return
     end

***** TERP ***************************************************************
**************************************************************************
***** cubic interpolation *************************************************
**************************************************************************
***** input ***************************************************************
* real*8                                                                  *
* p0, p1, p2 and p3 = samples (in order) at points spaced a unit          *
*                     distance apart.                                     *
* d2 = distance of test point from second sample point.                   *
**************************************************************************
***** output **************************************************************
* real*8                                                                  *
* terp = value at test point                                              *
**************************************************************************
***** common block ********************************************************
* APPROX consists of a single character variable that has value 'L'       *
* for Lagrange interpolation and 'S' for smooth jointed interpolation.    *
**************************************************************************
     real*8 function terp(d2,p0,p1,p2,p3)
     implicit real*8 (a-h,o-z)
     character app
     common /approx/app
      if(app.eq.'L') then
      d1=d2+1.
      d3=d2-1.
      d4=d2-2.
      terp=(p3*d1*d2*d3-p0*d2*d3*d4)/6.+(p1*d1*d3*d4-p2*d1*d2*d4)/2.
      else
      terp=p1+.5*(p2-p0)*d2+(p0-2.5*p1+2.*p2-.5*p3)*d2**2
    &+(.5*p3-1.5*p2+1.5*p1-.5*p0)*d2**3
      endif
     return
     end
```

```
***** DTERP ***********************************************************
************************************************************************
***** derivative of cubic interpolation ********************************
************************************************************************
***** input ************************************************************
* real*8                                                               *
* p0, p1, p2 and p3 = samples (in order) at points spaced a unit       *
*                     distance apart.                                  *
* d2 = distance of test point from second sample point.                *
************************************************************************
***** output ***********************************************************
* real*8                                                               *
* dterp = derivative at test point                                     *
************************************************************************
***** common block *****************************************************
* APPROX consists of single character variable that has value 'L'      *
* for Lagrange interpolation and 'S' for smooth jointed interpolation. *
************************************************************************
      real*8 function dterp(d2,p0,p1,p2,p3)
      implicit real*8 (a-h,o-z)
      character app
      common /approx/app
       if(app.eq.'L') then
       d1=d2+1.
       d3=d2-1.
       d4=d2-2.
       dterp=(p3*(d2*d3+d1*d3+d1*d2)-p0*(d3*d4+d2*d4+d2*d3))/6.
     &+(p1*(d3*d4+d1*d4+d1*d3)-p2*(d2*d4+d1*d4+d1*d2))/2.
        else
       dterp=.5*(p2-p0)+(2.*p0-5.*p1+4.*p2-p3)*d2
     &+(1.5*p3-4.5*p2+4.5*p1-1.5*p0)*d2**2
        endif
       return
       end

***** DET **************************************************************
************************************************************************
***** 3D determinant ***************************************************
************************************************************************
***** input ************************************************************
* real*8                                                               *
* a1(*) = 1st column of matrix                                         *
* a2(*) = 2nd column of matrix                                         *
* a3(*) = 3rd column of matrix                                         *
************************************************************************
***** output ***********************************************************
* real*8                                                               *
* det = determinant                                                    *
************************************************************************
      real*8 function det(a1,a2,a3)
      implicit real*8 (a-h,o-z)
      dimension a1(3),a2(3),a3(3)
      det=a1(1)*(a2(2)*a3(3)-a3(2)*a2(3))-a2(1)*(a1(2)*a3(3)
     &-a1(3)*a3(2))+a3(1)*(a1(2)*a2(3)-a1(3)*a2(2))
      return
      end
```

```
***** ELECTRONR ***************************************************************
*******************************************************************************
***** electron density and gradient for Chapman layer model ********************
*******************************************************************************
***** input ********************************************************************
* real*8                                                                       *
* r = distance from Earth centre                                               *
*******************************************************************************
***** output *******************************************************************
* real*8                                                                       *
* en = electron density (electrons per cubic cm)                               *
* diffr = derivative in vertical direction                                     *
*******************************************************************************
***** subroutines required *****************************************************
* CHAP, DCHAP and DET                                                          *
*******************************************************************************
***** common block *************************************************************
* DAT (defined in HASEL)                                                       *
*******************************************************************************
      subroutine electronr(r,en,diffr)
      implicit real*8 (a-h,o-z)
      dimension a1(3),a2(3),a3(3),b(3)
      common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
       if(r.lt.re) then
       eN=0.0
       diffr=0.0
       else
        if(max(foE,foF1).lt.1.d-7) then
         xN=foF2**2/80.6d-6
         eN=xN*chap(1.d0,1.4142d0*(r-re-hmF2)/ymF2)
         diffr=1.4142*xN*dchap(1.d0,1.4142d0*(r-re-hmF2)/ymF2)/ymF2
        else
         a1(1)=1.
         a2(1)=chap(.5d0,2.d0*(hmE-hmF1)/ymF1)
         a3(1)=chap(1.d0,1.4142d0*(hmE-hmF2)/ymF2)
         a1(2)=chap(.5d0,2.d0*(hmF1-hmE)/ymE)
         a2(2)=1.
         a3(2)=chap(1.d0,1.4142d0*(hmF1-hmF2)/ymF2)
         a1(3)=chap(.5d0,2.d0*(hmF2-hmE)/ymE)
         a2(3)=chap(.5d0,2.d0*(hmF2-hmF1)/ymF1)
         a3(3)=1.
         b(1)=foE**2/80.6d-6
         b(2)=foF1**2/80.6d-6
         b(3)=foF2**2/80.6d-6
         de=det(a1,a2,a3)
         c1=det(b,a2,a3)/de
         c2=det(a1,b,a3)/de
         c3=det(a1,a2,b)/de
         eN=c1*chap(.5d0,2.d0*(r-re-hmE)/ymE)
     &+c2*chap(.5d0,2.d0*(r-re-hmF1)/ymF1)
     &+c3*chap(1.d0,1.4142d0*(r-re-hmF2)/ymF2)
         diffr=2.*c1*dchap(.5d0,2.d0*(r-re-hmE)/ymE)/ymE
     &+2.*c2*dchap(.5d0,2.d0*(r-re-hmF1)/ymF1)/ymF1
     &+1.4142*c3*dchap(1.d0,1.4142d0*(r-re-hmF2)/ymF2)/ymF2
        endif
       endif
       return
       end

***** CHAP ********************************************************************
*******************************************************************************
***** chapman layer function ***************************************************
*******************************************************************************
***** input ********************************************************************
* real*8                                                                       *
* C = 1 in F2 layer                                                            *
*   = 1/2 in E and F1 layers                                                   *
* x = height above peak in units of scale height                               *
*******************************************************************************
***** output *******************************************************************
* real*8                                                                       *
* chap = value of Chapman function                                             *
*******************************************************************************
      real*8 function chap(C,x)
```

```
      implicit real*8 (a-h,o-z)
       xx=exp(-x)
      if(abs(xx).gt.1.d30) then
       chap=0.0
      else
       chap=exp(C*(1.d0-x-xx))
      endif
      return
      end

***** DCHAP ***********************************************************
**********************************************************************
***** derivative of chapman layer function ***************************
**********************************************************************
***** input ************************************************************
* real*8                                                              *
* C = 1 in F2 layer                                                   *
*   = 1/2 in E and F1 layers                                          *
* x = height above peak in units of scale height                      *
**********************************************************************
***** output ***********************************************************
* real*8                                                              *
* dchap = derivative of Chapman function                              *
**********************************************************************
      real*8 function dchap(C,x)
      implicit real*8 (a-h,o-z)
       xx=exp(-x)
      if(abs(xx).gt.1.d30) then
       dchap=0.0
      else
       dchap=C*(exp(-x)-1.d0)*exp(C*(1.d0-x-xx))
      endif
      return
      end

***** MAGNETIC ********************************************************
**********************************************************************
***** magnetic fields and derivatives for an offset dipole model *******
**********************************************************************
***** input ************************************************************
* real*8                                                              *
* x(*) = Cartesian coordinates (x(3) axis in the Earths axis of       *
*        rotation and x(1) axis in the plane of zero longitude).      *
***** output ***********************************************************
* real*8                                                              *
* v(*) = magnetic field components in Cartesian system                *
* ryx(*,*) = derivatives of manetic field components                  *
* ha = magnitude of magnetic field                                    *
* dip = magnetic dip angle                                            *
* dcl = declination of magnetic field                                 *
**********************************************************************
***** common block *****************************************************
* MAG (defined in HASEL)                                              *
**********************************************************************
      subroutine magnetic(x,v,ryx,ha,dip,dcl)
      implicit real*8 (a-h,o-z)
      dimension x(3),v(3),ryx(3,3)
      common /mag/fhs,fr,xm,ym,zm,px,py,pz,ir
      xmt=x(1)-xm
      ymt=x(2)-ym
      zmt=x(3)-zm
      pm=xmt*px+ymt*py+zmt*pz
      rr=xmt*xmt+ymt*ymt+zmt*zmt
      r=sqrt(rr)
      rp=1./(rr*r)
      v(1)=(px-3.*pm*xmt/rr)*rp
      v(2)=(py-3.*pm*ymt/rr)*rp
      v(3)=(pz-3.*pm*zmt/rr)*rp
      ra=sqrt(x(1)*x(1)+x(2)*x(2)+x(3)*x(3))
      ha=sqrt(v(1)*v(1)+v(2)*v(2)+v(3)*v(3))
      vr=v(1)*x(1)+v(2)*x(2)+v(3)*x(3)
      dip=-45.*asin(vr/(ha*ra))/atan(1.)
      dcl=45.*atan((v(2)*x(1)-v(1)*x(2))/(v(3)*ra-vr*x(3)))/atan(1.)
      ryx(1,1)=(-3.*pm/rr+6.*pm*xmt*xmt/(rr*rr)-3.*px*xmt/rr)*rp
```

```
     &-3.*xmt*(px-3.*pm*xmt/rr)*rp/rr
      ryx(1,2)=(-3.*py*xmt/rr+6.*pm*ymt*xmt/(rr*rr))*rp
     &-3.*ymt*(px-3.*pm*xmt/rr)*rp/rr
      ryx(1,3)=(-3.*pz*xmt/rr+6.*pm*zmt*xmt/(rr*rr))*rp
     &-3.*zmt*(px-3.*pm*xmt/rr)*rp/rr
      ryx(2,2)=(-3.*pm/rr+6.*pm*ymt*ymt/(rr*rr)-3.*py*ymt/rr)*rp
     &-3.*ymt*(py-3.*pm*ymt/rr)*rp/rr
      ryx(2,3)=(-3.*pz*ymt/rr+6.*pm*zmt*ymt/(rr*rr))*rp
     &-3.*zmt*(py-3.*pm*ymt/rr)*rp/rr
      ryx(3,3)=(-3.*pm/rr+6.*pm*zmt*zmt/(rr*rr)-3.*pz*zmt/rr)*rp
     &-3.*zmt*(pz-3.*pm*zmt/rr)*rp/rr
      ryx(2,1)=ryx(1,2)
      ryx(3,2)=ryx(2,3)
      ryx(3,1)=ryx(1,3)
      return
      end

***** FUNC ***************************************************************
*************************************************************************
***** right hand side of Haselgrove equations ***************************
*************************************************************************
***** input **************************************************************
* real*8                                                                 *
* vec(*) = value of solution vector at which right hand required         *
* integer                                                                *
* n = number of components in vectors vec and f                          *
***** output *************************************************************
* real*8                                                                 *
* f(*) = right hand side of Haselgrove equations                         *
*************************************************************************
***** subroutines required ***********************************************
* ELECTRON, MAGNETIC                                                     *
*************************************************************************
***** common block *******************************************************
* MAG (defined in HASEL)                                                 *
*************************************************************************
***** notes **************************************************************
* In the current code (HASEL), y has nine components. In order, these    *
* are:                                                                   *
*     the Cartesian coordinates at the current point,                    *
*     the components of the wave vector,                                 *
*     the group path,                                                    *
*     the Doppler shift,                                                 *
* and the phase path.                                                    *
*************************************************************************
      subroutine func(vec,f,n)
      implicit real*8 (a-h,o-z)
      dimension f(n),vec(n),v(3),ryx(3,3),dnx(4)
      common /mag/fhs,fr,xm,ym,zm,px,py,pz,ir
      call electron(vec,en,dnx)
      rxt=8.06d-5*dnx(4)/(fr*fr)
      rx=8.06d-5*en/(fr*fr)
      vv=vec(4)**2+vec(5)**2+vec(6)**2
       if(abs(fhs).gt.1.0e-20) then
         call magnetic(vec,v,ryx,ha,dip,dcl)
         vu=(vec(4)*v(1)+vec(5)*v(2)+vec(6)*v(3))/ha
         ry=ha
         ry2=ry*ry
         ga=-vu**2
         gg=-ga/vv
         al=1.-rx-ry2
         be=.5*ry*(1.-ga)
         fac=sqrt(abs(be**2-al*ga))
          if(ir.eq.1) then
           rp=-ga/(be+fac)
           rj=4.*fac
          else
           rp=-(be+fac)/al
           rj=-4.*fac
          endif
         rp2=rp**2
         rk=-2.*rx*vu*(rp*ry-1.)
         rl=ry*(1.-ry2)*rp2-2.*(1.-rx-ry2)*rp-ry
         rm=2.*rx*rp*(rp*ry-1.)
```

```
          do i=1,3
           f(i)=rj*vec(i+3)-rk*v(i)
           dxx=8.06e-5*dnx(i)/(fr*fr)
           f(i+3)=rl*dxx+(rk*vec(4)+rm*v(1))*ryx(1,i)
     &+(rk*vec(5)+rm*v(2))*ryx(2,i)+(rk*vec(6)+rm*v(3))*ryx(3,i)
          enddo
         rx1=1.-rx
         a=a1+rx*ry2*gg
         b=rx1*a1-.5*rx*ry2*(1.-gg)
         c=rx1*(rx1**2-ry2)
         emu=(b+ir*sqrt(abs(b**2-a*c)))/a
         eml=emu*a-b
         ems=sqrt(abs(emu))
          if(abs(eml).gt.1.0d-9) then
           aa=1.-1.5*rx-1.5*ry2+2.*rx*ry2*gg
           bb=rx1*(1.-3.*rx)-2.*ry2+1.5*rx*ry2*(1.+gg)
           cc=-1.5*rx*rx1**2-ry2*(.5-rx)
           emu=(aa*emu*ems-bb*ems+cc/ems)/eml
           ax=-1.+gg*ry2
           bx=-2.*rx1+.5*ry2*(1.+gg)
           cx=-3.*rx1**2+ry2
           emx=.25*(-ax*ems**4+2.*bx*ems**2-cx)/(a*ems**3-b*ems)
          else
           emu=1./sqrt(abs(1.-rx))
           emx=-.5*emu
          endif
        else
         ems=sqrt(abs(1.-rx))
          do i=1,3
           f(i)=vec(i+3)
           dxx=8.06e-5*dnx(i)/(fr*fr)
           f(i+3)=-.5*dxx
          enddo
         emu=1./ems
         emx=-.5*emu
        endif
         cosa=(vec(4)*f(1)+vec(5)*f(2)+vec(6)*f(3))/sqrt(vv)
         f(7)=emu*cosa
         f(8)=-cosa*fr*rxt*emx/3.e5
         f(9)=ems*cosa
        if(abs(fhs).gt.1.0e-20) then
         do i=1,n
          f(i)=real(ir)*f(i)
         enddo
        endif
       return
       end

***** RKF ********************************************************************
*******************************************************************************
***** Runge-Kutta-Fehlberg scheme to solve dy/dt= f ***************************
*******************************************************************************
***** input *******************************************************************
* real*8                                                                      *
* t = initial value of solution parameter (changed on exit)                   *
* y(*) = initial value of solution vector (changed on exit)                   *
* h = initial estimate of change in t                                         *
* tol = maxim error that can be tolerated in this step                        *
* hmin= minimum h that can be tolerated                                       *
* hmax= maximum h that can be tolerated                                       *
* integer                                                                     *
* n = number of components in vector y                                        *
***** output ******************************************************************
* real*8                                                                      *
* t = exit value of solution parameter                                        *
* y(*) = estimate of solution vector at exit t                                *
* h = estimate of h for next step                                             *
*******************************************************************************
***** subroutine required *****************************************************
* FUNC                                                                        *
* This calculates the value of f corresponding to y. There are three          *
* arguments. The first is y (real*8), the second is f (real*8) and            *
* the third is the number of components in arrays y and f (integer).          *
*******************************************************************************
```

```
      subroutine rkf(t,y,n,h,tol,hmin,hmax)
      implicit real*8 (a-h,o-z)
      dimension a1(11),a2(11),a3(11),a4(11),a5(11),a6(11),y(n),yt(11)
      iend=0
 99   iend=iend+1
      call func(y,a1,n)
      alp=0.0
       do i=1,n
       alp=alp+a1(i)**2
       yt(i)=y(i)+.25*a1(i)*h
       enddo
      alp=max(sqrt(abs(alp)),1.0d-32)
      call func(yt,a2,n)
       do i=1,n
       yt(i)=y(i)+(.09375*a1(i)+.28125*a2(i))*h
       enddo
      call func(yt,a3,n)
       do i=1,n
       yt(i)=y(i)+(.87938097406d0*a1(i)-3.2771961766d0*a2(i)
     &+3.32089212563d0*a3(i))*h
       enddo
      call func(yt,a4,n)
       do i=1,n
       yt(i)=y(i)+(2.0324074074d0*a1(i)-8.*a2(i)+7 17348927875d0*a3(i)
     &-.20589668616d0*a4(i))*h
       enddo
      call func(yt,a5,n)
       do i=1,n
       yt(i)=y(i)+(-.29629629629d0*a1(i)+2.*a2(i)-1.38167641326d0*a3(i)
     &+.45297270955d0*a4(i)-.275*a5(i))*h
       enddo
      call func(yt,a6,n)
      erx=0.0
       do i=1,3
       erz=.277777777777778d-2*a1(i)-.299415204678363D-1*a3(i)
     &-.291998936735779D-1*a4(i)+.2D-1*a5(i)+.363635363636364D-1*a6(i)
       erx=erx+erz**2
       enddo
      erx=sqrt(erx)
      hold=h
       if(erx.gt.0.001*tol) then
       s=.84*sqrt(sqrt(abs(tol/erx)))
       h=s*h
       else
       h=2.*h
       endif
      if(h.lt.hmin/alp) h=hmin/alp
      if(h.gt.hmax/alp) h=hmax/alp
       if(erx.gt.tol.and.iend.lt.8) then
       go to 99
       else
        do i=1,n
         y(i)=y(i)+(.11851851852*a1(i)+.51898635478d0*a3(i)
     &+.50613149034d0*a4(i)-.18*a5(i)+.03636363636*a6(i))*hold
        enddo
       t=t+hold
       return
       endif
      end
```

```
***** ELDEN ********************************************************************
*******************************************************************************
***** general electron density *************************************************
*******************************************************************************
***** input ********************************************************************
* real*8                                                                       *
* t = time from start (sec)                                                    *
* elong = longitude (deg) of sample point                                      *
* elat = latitude (deg) of sample point                                        *
* r = height above Earth centre (Kms)                                          *
*                                                                              *
* Note that hmE,hmF1 and hmF2 should be set to suitable values to              *
* allow the labelling of rays                                                  *
***** output *******************************************************************
* real*8                                                                       *
* elden = electron density  (electrons per cubic cm)                           *
*******************************************************************************
***** common block (optional) *************************************************
* DAT (defined in HASEL)                                                       *
*******************************************************************************
      real*8 function elden(t,elong,elat,r)
      implicit real*8 (a-h,o-z)
      common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
      foF2=12.
      hmF2=350.
      ymF2=110.
      vel=.04
      x=(r-re-hmF2-vel*t)/ymF2
      elden=(foF2**2/80.6d-6)*chap(1.d0,1.4142d0*x)+0.*elong*elat
      return
      end

*******************************************************************************
*******************************************************************************
*******************************************************************************
```

# DISTRIBUTION

Page No.

**Defence Science and Technology Organisation**

Chief Defence Scientist )
Central Office Executive ) 1

Counsellor, Defence Science, London Cnt Sht Only

Counsellor, Defence Science, Washington Cnt Sht Only

Scientific Adviser, Defence Central 1

**Navy Office**

Navy Scientific Adviser 1

**Air Office**

Air Force Scientific Adviser 1

**Army Office**

Scientific Adviser, Army 1

**Defence Intelligence Organisation**

Scientific Adviser 1

**Surveillance Research Laboratory**

Director, Surveillance Research Laboratory 1

Chief High Frequency Radar Division 1

Research Leader, Jindalee Operational Radar Network 1

Head, Radar Processing and Tracking 1

Head, Ionospheric Effects 1

Head, Radar Technology and Systems 1

Head, HF Radar Engineering 1

Graphics and Documentation, High Frequency Division 1

Dr J.A Bennett, Ionospheric Effects 1

**DISTRIBUTION** (Contd)

| | |
|---|---|
| Chief Microwave Radar Division | Cnt Sht Only |
| Research Leader Microwave Radar | 1 |
| Research Leader Radar Projects | 1 |
| Author | 1 |

**Electronics Research Laboratory**

| | |
|---|---|
| Chief Communications Division | Cnt Sht Only |
| Research Leader Military Communications | 1 |
| Research Leader Communications Countermeasures | 1 |
| Dr K.J.W Lynn, Radiowave Propagation | 1 |
| Dr R.H Clarke, Radiowave Propagation | 1 |

**Libraries and Information Services**

| | |
|---|---|
| Australian Government Publishing Service | 1 |
| Defence Central Library - Technical Reports Centre | 1 |
| Manager, Document Exchange Centre (For Retention) | 1 |
| National Technical Information Service, United States | 2 |
| Defence Research Information Centre, United Kingdom | 2 |
| Director Scientific Information Services, Canada | 1 |
| Ministry of Defence, New Zealand | 1 |
| National Library of Australia | 1 |
| Defence Science and Technology Organisation Salisbury, Research Library | 2 |
| British Library Document Supply Centre | 1 |
| Library Defence Signals Directorate, Melbourne | 1 |

**Spares**

| | |
|---|---|
| Defence Science and Technology Organisation Salisbury, Research Library | 6 |

Department of Defence

# DOCUMENT CONTROL DATA SHEET

| Page Classification |
|---|
| UNCLASSIFIED |
| Privacy Marking/Caveat (of Document) |
| N/A |

| 1a. AR Number | 1b. Establishment Number | 2. Document Date | 3. Task Number |
|---|---|---|---|
| AR-008-171 | SRL-0131-TR | August 1993 | DST 91/027 |

**4. Title**

A GENERAL PURPOSE IONOSPHERIC RAY TRACING PROCEDURE

**5. Security Classification**

| Document | Title | Abstract |
|---|---|---|
| U | U | U |

S (Secret) C (Confi ) R (Rest) U (Unclass)

* For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.

| 6. No. of Pages | 34 |
|---|---|
| 7. No. of Refs. | 5 |

**8. Author(s)**

C.J. Coleman

**9. Downgrading/Delimiting Instructions**

N/A

**10a. Corporate Author and Address**

Surveillance Research Laboratory
PO Box 1500
SALISBURY SA 5108

**10b. Task Sponsor**

**11. Officer/Position responsible for**

Security.....N/A.....

Downgrading.....N/A.....

Approval for Release.....DSRL.....

**12. Secondary Distribution of this Document**

APPROVED FOR PUBLIC RELEASE

Any enquiries outside stated limitations should be referred through DSTIC, Defence Information Services, Department of Defence, Anzac Park West, Canberra, ACT 2600.

**13a. Deliberate Announcement**

No limitation

**13b. Casual Announcement (for citation in other documents)**

[√] No Limitation

[ ] Ref. by Author , Doc No. and date only.

**14. DEFTEST Descriptors**

Ionospheric propagation, Ionospheric properties, Ray tracing, Computer applications

**15. DISCAT Subject Codes**

0401, 1205

**16. Abstract**

A general purpose ionospheric ray tracing procedure is developed. The procedure is based on a numerical solution to the Haselgrove ray tracing equations and includes calculations of ionospherically induced Doppler shift. The procedure can handle a wide variety of ionospheric descriptions and includes the effect of the Earth's magnetic field.

| Page Classification |
| --- |
| UNCLASSIFIED |

16. Abstract (CONT.)

17. Imprint

Surveillance Research Laboratory
PO Box 1500
SALISBURY SA 5108

| 18. Document Series and Number | 19. Cost Code | 20. Type of Report and Period Covered |
| --- | --- | --- |
| SRL-0131-TR | 440/828309 | TECHNICAL REPORT |

21. Computer Programs Used

FORTRAN

22. Establishment File Reference(s)

N/A

23. Additional information (if required)